(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 April 2002 (11.04.2002)

PCT

(10) International Publication Number
WO 02/30044 A2

(51) International Patent Classification[7]: H04L 12/00

(21) International Application Number: PCT/US01/30112

(22) International Filing Date:
25 September 2001 (25.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/680,022 5 October 2000 (05.10.2000) US

(71) Applicant: **WIND RIVER SYSTEMS, INC.** [US/US]; 500 Wind River Way, Alameda, CA 94501 (US).

(72) Inventors: **WILES, Roger**; 2907 Yukon Drive, Corinth, TX 76210 (US). **IRWIN, Thomas, Dean**; 3904 Hidden Trail, Flower Mound, TX 75002 (US).

(74) Agents: **FAY, Patrick, J.** et al.; Fay Kaplun & Marcin, LLP, 17th Floor, 100 Maiden Lane, New York, NY 10038 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**
— *without international search report and to be republished upon receipt of that report*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

WO 02/30044 A2

(54) Title: A SYSTEM AND METHOD FOR IMPLEMENTING MULTI-LEVEL NETWORK DRIVERS

(57) Abstract: A software package for operating on a network hardware device having a plurality of physical ports, comprising at least one networking protocol to transmit and receive data packets, a plurality of first network drivers communicatively coupled to the plurality of physical ports of the network hardware device, a plurality of second network drivers selectively communicatively coupleable to a first one of the plurality of first network drivers and selectively communicatively coupleable to a second one of the plurality of first network drivers, and a plurality of third network drivers, each third network driver being associated with a corresponding VLAN, wherein each of the third network drivers is selectively communicatively coupleable to at least one of the second network drivers that is a member of the corresponding VLAN, and wherein data packets for transmission by the network hardware device travel from communicatively coupled ones of the third network drivers to the corresponding second network drivers, from communicatively coupled ones of the second network drivers to the first network drivers communicatively coupled thereto and from communicatively coupled ones of the first network drivers to the physical ports to which the communicatively coupled ones of the first network driver are coupled.

# A SYSTEM AND METHOD FOR IMPLEMENTING MULTI-LEVEL NETWORK DRIVERS

## Background Information

A computer network is simply a group of two or more computers that are linked together. Many types of networks exist, but the most common types of networks are Local-Area Networks (LANs) and Wide-Area Networks (WANs). In a LAN computers are connected within a "local" area, for example, a home or office. A LAN in a home or small office may interconnect a few computers, whereas in the case of a large office or industrial complex, the LAN may include hundreds or even thousands of interconnected computers. In a WAN, the interconnected computers are generally farther apart and are connected via telephone/communication lines, radio waves, or other means of communication. There are also other network devices, for example, bridges, routers and switches, which may be used to link different networks or network segments into a larger network with more resources.

Each of these networks and network segments may have a peer-to-peer arrangement where each computing device is interconnected with the others on the network, or in a client/server arrangement where individual personal computers (PCs) log onto a server device. Additionally, a network is not limited to computers, but may also include other types of hardware devices and information services, for example, the bridges, routers and switches described above, printers, copiers and shared Internet connections. A network allows each of the users to interconnect with the other devices and services on the network for purposes such as application and file sharing, printer access, electronic mail and Internet access.

In a heterogeneous environment, such as a computer network, it is essential that each of the interconnected devices be able to receive information from and transmit information to the other devices in the network. The information transferred between network devices is generally referred to as data packets or packets and this transfer of information is generally referred to as packet flow. In response to the need for the different network devices to communicate, standards bodies and other entities have developed numerous protocols for data packet flow over a network. A protocol specifies a common set of rules for packet

format and packet flow, allowing a variety of devices to communicate. Generally, a data packet contains a destination address that is associated with one or more of the network devices. As the data packet travels through the network, the network routing and/or switching devices are able to target the data packet to the correct destination based on the address.

Some networks implement Virtual Local Area Network (VLAN) technology. VLANs allow a network administrator to have separate LANs among ports on the same switch or router. For example, a switch with 16 ports may be separated into two VLANs, ports 1-8 assigned to VLAN(A) and ports 9-16 assigned to VLAN(B). A data packet that is addressed for broadcast to VLAN(A) would only be sent out via the ports assigned to VLAN(A), e.g. ports 1-8. Similarly, a data packet that is addressed for broadcast to VLAN(B) would only be sent out via the ports assigned to VLAN(B), e.g. ports 9-16. Thus, a VLAN is simply a territory over which a broadcast packet is to be delivered, sometimes referred to as a broadcast domain.

There are numerous reasons why a network administrator would choose to implement a VLAN. For example, a VLAN allows the isolation of users groups so that their broadcast traffic will not interfere with that of other users groups. For example, a marketing group and a sales group may share a single network switch. However, the marketing group may be assigned to a first VLAN and the sales group may be assigned to a second VLAN. Thus, the broadcast traffic from the marketing group will not interfere with the sales group and vice versa. Additionally, some protocols are overly chatty, or they can get into modes such as broadcast storms. Therefore, keeping protocols on separate VLANs can protect one protocol from the bad behavior of another.

Network software generally assigns a single network driver for each network device. Therefore, when a switch, or other network device, has multiple ports assigned to different VLANs, the network driver is not able to distinguish between the different VLANs. Thus, the single network driver is not capable of directing data packets to the individual VLANs. Similarly, when a single port is a member of multiple VLANs and it receives a data packet,

the network driver does not know over which VLAN the data packet is associated with and therefore may need to send the data packet to all ports in each of the VLANs that the receiving port is a member. This flooding of data packets to more ports than necessary defeats some of the primary purposes of implementing VLANs.

## Summary of the Invention

A software package for operating on a network hardware device having a plurality of physical ports. The software package has at least one networking protocol to transmit and receive data packets, a plurality of first network drivers communicatively coupled to the plurality of physical ports of the network hardware device, a plurality of second network drivers selectively communicatively coupleable to a first one of the plurality of first network drivers and selectively communicatively coupleable to a second one of the plurality of first network drivers, and a plurality of third network drivers, each third network driver being associated with a corresponding VLAN. Each of the third network drivers is selectively communicatively coupleable to at least one of the second network drivers that is a member of the corresponding VLAN. Data packets for transmission by the network hardware device travel from communicatively coupled ones of the third network drivers to the corresponding second network drivers, from communicatively coupled ones of the second network drivers to the first network drivers communicatively coupled thereto and from communicatively coupled ones of the first network drivers to the physical ports to which the communicatively coupled ones of the first network driver are coupled.

## Brief Description of Drawings

Fig. 1 shows a network switch having its ports divided into a first exemplary VLAN configuration.

Fig. 2 shows a network switch having its ports divided into a second exemplary VLAN configuration.

Fig. 3 shows an exemplary linking of network devices.

Fig. 4 shows a block diagram of a first exemplary embodiment of stacked network drivers according to the present invention.

Fig. 5 shows an exemplary connection of network drivers to multiple subnets according to the present invention.

Fig. 6 shows a block diagram of a second exemplary embodiment of stacked network drivers according to the present invention.

Fig. 7 shows a block diagram of a third exemplary embodiment of stacked network drivers according to the present invention.

Fig. 8 shows an exemplary flow process for a data packet through upper level protocols and stacked network drivers according to the present invention.

**Detailed Description**

The present invention may be further understood with reference to the following description and the appended drawings, wherein like elements are provided with the same reference numerals. The following description describes exemplary embodiments of the present invention with references to switches, however, those skilled in the art will recognize that the present invention may be implemented on any network device, for example routers, servers, work stations, etc. Fig. 1 shows exemplary switch 20 having ports 1-12 that are assigned to VLANs 21-23. As described above, a VLAN provides a mechanism to subdivide a network into multiple logical networks or broadcast domains. VLAN 21 includes member ports 1, 2 and 4, VLAN 22 has ports 3, 5, 6 and 7 and VLAN 23 has ports 8-12. This grouping of the ports into VLANs restricts network traffic to a subset of the entire network, conserving network bandwidth. Broadcasts are only sent to the VLAN member ports. Normally, a broadcast is a data packet that is addressed for destination to all nodes in the network. If ports 1-12 of switch 20 were not divided into separate VLANs, then a broadcast data packet would be sent out all of ports 1-12. However, since switch 20 is divided into separate VLANs, a broadcast data packet received on any one of the ports will be sent out only on the

other ports in that VLAN. For example, a broadcast packet received on port 11 would only be sent out on ports 8-10 and 12, the other ports on VLAN 23.

Similarly, unknown unicast data packets are only flooded to the VLAN member ports. A unicast is a data packet that is addressed for destination to a single node. However, in certain instances the switch will not know the port that is associated with the addressed node. In this case, the switch will flood the unicast data packet to all the ports in the VLAN to make sure that correct node receives the data packet, but it does not need to flood the data packet to all ports of switch 20. For example, if an unknown unicast address is received on port 4, it is flooded only to ports 1 and 2, the other ports of VLAN 21.

Fig. 2 also shows network switch 20, but in this exemplary embodiment ports 1-12 are configured differently from Fig. 1. Ports 1-4 are assigned to VLAN 31, ports 4-7 are assigned to VLAN 32, ports 4 and 8-12 are assigned to VLAN 33 and ports 2, 6 and 10 are assigned to VLAN 34. As can be seen from Fig. 2, it is possible to assign a single port to more than one VLAN. For example, port 4 is assigned to VLANs 31, 32 and 33. Device 40 attached to port 4, may be, for example, a file server that each of the users on VLANs 31-33 may wish to access. Therefore, file server 40 should be available to each of VLANs 31-33. Data packet addressing allows a single port to be assigned to more than one VLAN. One example of data packet addressing allowing this configuration is given by IEEE Standard 802.1Q Virtual Bridged Local Area Networks. In this method of addressing, each data packet contains a 4 byte tag. The first two bytes identifies the packet as an 802.1Q packet and the last two bytes are broken down into 3 bit fields that include a 3 bit priority, a Canonical Format Indicator (e.g., 0 for Ethernet, 1 for token ring, etc), and a 12 bit VLAN identifier (VID). This 12 bit VID allows a data packet to contain information about the particular VLAN for which the data packet is targeted. For example, file server 40 may send a broadcast packet for VLAN 32. This broadcast packet would have a VLAN identifier signaling that it is destined for VLAN 32. When the data packet is received by port 4 of switch 20, the VLAN identifier is recognized by the switch and the data packet is broadcast to only ports 5-7, the other ports of VLAN 32. Those skilled in the art will understand that the IEEE 802.1Q addressing is only used for exemplary purposes and that there are numerous other addressing schemes that may

be used to identify a VLAN target for a data packet.

Link aggregation is a technique that may be used by itself or in conjunction with VLANs to increase link availability and bandwidth between network devices by allowing multiple physical links to be combined to operate as a single logical link. Fig. 3 illustrates an example of link aggregation between three network devices, in this case network switches 100 and 110 and file server 120. Network switch 100 has ports 101-106, network switch 110 has ports 111-116 and file server 120 has ports 121-123. Ports 101, 102 and 104 of network switch 100 are physically linked to ports 111, 112 and 113 of network switch 110, respectively, via links 131-133. Ports 103, 105 and 106 of network switch100 are physically linked to ports 121, 122 and 123 of file server 120, respectively, via links 134-136. Those skilled in the art will understand that the physical links may be any physical medium over which a data packet may be transferred, for example, fiber optic cable, coaxial cable, twisted pairs, etc. Link aggregation allows these physical links between network devices to be aggregated into a single logical link resulting in increased bandwidth and fault tolerance between the network devices, for example, switches, routers and file servers.

For example, physical links 131-133 between network switch 100 and network switch 110 may be aggregated into a single logical link. If, for example, each of ports 101, 102, 104, 111, 112 and 113 were 100 Mbps ports operating in full duplex mode, the single logical link resulting from physical links 131-133 would provide a bandwidth between network switches 100 and 110 to 600 Mbps. Full duplex mode means that each network port may simultaneously transmit and receive data packets over a point to point link. Since each port can simultaneously transmit and receive data, the throughput of the link is effectively doubled. A 100 Mbps port operating in full duplex mode provides a maximum bandwidth of 200 Mbps. Therefore, the aggregation of three physical links into a single logical link results in an effective bandwidth of 600 Mbps.

Link aggregation also provides additional fault tolerance because a fault on any single link will not inhibit all traffic between the two stations. With link aggregation, traffic on a failing link can be redirected to one of the other links in the group. If, for example, link 131 between

network switches 100 and 110 is broken or interrupted, data packets may still flow between network switches 100 and 110 via the remaining aggregated links, *i.e.*, links 132 and 133. Those skilled in the art will understand that it may not be possible to aggregate all physical links between network devices because of the operating parameters of the ports. For example, if links 134 and 135 between network switch 100 and filer server 120 are operating at 100 Mbps and link 136 between those same network devices is operating at 10 Mbps, it may not be possible to aggregate link 136 with links 134 and 135 because of the difference in operating speeds.

With link aggregation it is also possible to perform load balancing by distributing traffic across the multiple links. The load balancing algorithm may evenly distribute the traffic among the links, or guarantee bandwidth by dedicating one or more links to high priority traffic. Load balancing, or determining the physical port that should actually transmit the data packet, will be discussed in greater detail below. Those skilled in the art will recognize that link aggregation may be used to link any number of physical ports up to the maximum number of ports on the device. For example, in Fig. 3, all six ports 101-106 of network switch 100 may be physically connected to ports 111-116 of network switch 110. These six physical links may form a single logical link.

Some network devices may implement both VLANs and link aggregation. Fig. 4 shows a block diagram of an exemplary embodiment of stacked network drivers for hardware device 240 according to the present invention. Those skilled in the art will understand that the embodiment of the present invention that will be described below does not require that the network implement VLANs or link aggregation. The network may implement one of, neither or both of VLANs and link aggregation. Additionally, those skilled in the art will recognize that the network drivers will be described as residing in the overall structure of the network software, but it is not a requirement that one or more of the network drivers reside in the network software. The network drivers may also be one or more interfaces separate from the network software. Those skilled in the art will also understand that the present invention may be implemented in hardware devices such as an application specific integrated circuit (ASIC), network processors, state machines, etc.

As described above, a protocol specifies a common set of rules for packet format and packet flow. This protocol is implemented via the network software which is resident on various network hosts which may include the connected hardware devices. The network software controls the packet flow through the network. Throughout this specification the terms network software and protocol may be used interchangeably to describe the implementation of this common set of rules for packet format and packet flow. Each of the blocks in Fig. 4, excluding network hardware device 240, may be considered a network driver. The network driver is the interface between the network software, specifically the upper level layers and applications, and the hardware device. In addition, to the specific nomenclature used to identify these blocks in Fig. 4, the term network drivers will be used to generically refer to any one of these blocks or to multiples of these blocks. The function of each of these network drivers will be described in greater detail below. Fig. 4 does not show any upper level layers or applications of the network software with which the network drivers interface. A more complete embodiment, including these items, will be described below.

Fig. 4 shows four levels 200-230 of stacked network drivers implemented on network switch 240 which has eight physical ports 241-248. Each of the network drivers binds to the levels immediately above and below, and functions to assist in simplifying the network drivers to which it is bound. Binding refers to a physical, dynamic, or logical linking of layers through functional interfaces of other techniques. These links provide a connection through which data and control information may be communicated between layers. Level 200 has three interface network drivers 201-203 that bind in the upward direction to the upper level layers of the network software (not shown) and bind in the downward direction to the level 210 network drivers. Level 210 has eight aggregator drivers 211-218 that bind in the upward direction to the level 200 network drivers and in the downward direction to the level 220 network drivers. Level 220 has eight aggregation port drivers 221-228 that bind in the upward direction to the level 210 network drivers and in the downward direction to the level 230 network drivers. Lastly, level 230 has eight hardware device drivers 231-238 that bind in the upward direction to the level 220 network drivers and in the downward direction to ports 241-248 of the actual hardware device, network switch 240. Data packet flow through network drivers of levels 200-230 will be described in greater detail below. However, in

general, when level 230 hardware device drivers 231-238 receive an incoming data packet it is passed to the level 220 aggregation port drivers 221-228 which passes the data packet to the level 210 aggregator drivers 211-218, which passes the data packet to the level 200 interface network drivers 201-203, which passes the data packet to the upper level layers of the network software. Those skilled in the art will understand that the four levels of network drivers illustrated in Fig. 4 are only exemplary and that the present invention may be implemented using any number of stacked network drivers.

Level 200 interface network drivers 201-203 provide a generic interface between the upper level layers (not shown) and the underlying network drivers, for example, level 210 network drivers. Each of interface network drivers 201-203 is associated with a single VLAN. In the example of Fig. 4, the network devices connected to ports 241-248 of switch 240 are members of three separate VLANs, VLAN(A), VLAN(B), and VLAN(C). Therefore, there are three interface network drivers 201-203, with interface network driver 201 assigned to VLAN(A), interface network driver 202 assigned to VLAN(B), and interface network driver 203 assigned to VLAN(C). If on the other hand, each of ports 241-248 was assigned to a different VLAN, there would be eight interface network drivers. Because each interface network driver 201-203 is assigned to only one VLAN, it receives a unique VID so that data packets containing this VID are targeted to the correct interface network driver.

As shown in Fig. 4 and as will be discussed in greater detail below, level 210 network drivers may be members of more than one VLAN, and therefore, may bind to more than one level 200 interface network driver. When a level 210 network driver receives a data packet from a lower level network driver, it must determine which of the level 200 interface network drivers to send the data packet to. The level 210 network driver will read the VID contained in the data packet and associate the data packet VID with a VID of the level 200 interface network driver to which the data packet should be sent. For example, when aggregator 214 receives a data packet, it may send it to interface network driver 201 on VLAN(A) or interface network driver 202 on VLAN(B). Aggregator 214 learns each of the VIDs when it binds to interface network drivers 201 and 202. Aggregator 214 will then be able to perform a lookup on the VID in the data packet and forward the data packet to the correct one of interface network

drivers 201 and 202 based on the VIDs.

Each interface network driver 201-203 may be associated with more than one subnet, but may be associated with only a single VLAN. Subnetting is a technique that subdivides a physical network into multiple logical networks. It is most commonly used in networks implementing IP (Internet Protocol Suite), but it is not limited to that protocol suite. Subnetting a network may be done for a variety of reasons, including organization, use of different physical media (such as Ethernet, FDDI, WAN, etc.), preservation of address space, and security. Fig. 5 shows interface network drivers 201-203, described with respect to Fig. 4, connected in an exemplary manner to subnets 251-254. As described above, the number of interface network drivers is equal to the number of VLANs. However, as shown in Fig. 5, it is possible for an interface network driver to be connected to multiple subnets. For example, interface network driver 201 is connected to subnets 251 and 252, whereas each of interface network drivers 202 and 203 is connected to a single subnet; subnets 253 and 254, respectively.

The upper level layers of the network software, for example, IP, may not be able to understand identifications of port numbers or a particular VLAN on which a data packet is to be transmitted, but only understand identifications of subnets. Those skilled in the art will understand that there are some upper level applications, for example, STP and GARP, that may be able to understand port numbers. The level 200 interface network drivers 201-203 provide the interface between the subnets and the VLANs or ports. As described above, a VLAN is simply a collection of ports that allows the subdivision of a network into multiple logical networks or broadcast domains. Interface network drivers 201-203 contain information about the level 210 network drivers which allow data packets to be addressed so that they are sent to the correct level 210 network drivers. For example, the upper level layers may have a data packet that is to be sent on subnet(C) 253, but these upper level layers do not know which VLAN or ports are associated with subnet(C) 253. The data packet is forwarded to interface network driver 202 which is connected to subnet(C) 253 and addressed based on the information contained in the interface network driver 202 about the level 210 network drivers. As a further example that is specific to an IP network, interface network drivers 201-

203 may contain an address resolution table (ARL) that is used to perform a correlation between the subnet and the VLAN. IP may provide the ARL with a destination MAC (Medium Access Control) address that is associated with a particular subnet. The ARL may then correlate this MAC address to a particular VLAN and aggregator driver number.

Referring back to Fig. 4, level 210 aggregator drivers 211-218 represent the logical ports of network switch 240. A logical port represents a link over which network switch 240 may send or receive data packets and may correspond to any number of physical ports, including zero. For example, in Fig. 4, aggregator drivers 212, 216 and 218 are logical ports, however, in this configuration they are not associated with any of physical ports 241-248. Whether a level 210 aggregator driver is ultimately bound to one physical port, *e.g.*, aggregator driver 213, or bound to multiple physical ports, *e.g.*, aggregator driver 215, it is considered a single logical port over which data packets may be sent or received. When aggregator drivers 211-218 bind to the level 200 interface network drivers, information about the aggregator drivers is relayed to the level 200 interface network drivers so that packets may be correctly addressed. For example, aggregator drivers 211-214 are bound to interface network driver 201 which is assigned to VLAN(A). During the binding process, information about each of aggregator drivers 211-214 is stored in interface network driver 201. This information may also be stored as an aggregator driver dynamically changes its binding to another interface network driver. An aggregator driver may bind to multiple interface network drivers allowing each aggregator driver to be a member of multiple VLANs. For example, aggregator driver 214 is bound to interface network drivers 201 and 202, and therefore is a member of both VLAN(A) and VLAN(B).

In the example of Fig. 4, there is a one-to-one ratio of aggregator drivers 211-218 to physical ports 241-248 on switch 240. There is no requirement that there be an equal number of aggregator drivers to physical ports. For example, aggregator drivers 212, 216 and 218 are bound in the upward direction to network drivers in level 200, but are not bound in the downward direction to any network drivers in level 220. This means that aggregator drivers 212, 216 and 218 are not connected through the lower levels to the actual hardware ports. Thus, a data packet received by switch 240 does not have a path, in this configuration,

through aggregator drivers 212, 216 and 218. Similarly, on data packet transmission, if a data packet were sent to aggregator drivers 212, 216 and 218, it would never leave switch 240 because there is no path to pass the data packet to a lower level driver and ultimately out a physical port of switch 240. Therefore, aggregator driver 212, being a member of VLAN(A), will receive the broadcast traffic or flood traffic designated for VLAN(A), but it will simply discard the data because it is not connected to any of aggregation port drivers 221-228. There may also be some upper level applications and application protocols, for example, STP (Spanning Tree Protocol), GARP (Generic Attribute Registration Protocol), etc., which have the ability to recognize ports and bind directly to the aggregator drivers. In these cases, the applications would recognize aggregator drivers 212, 216 and 218 as logical ports, but would consider them to be in an "off" or "down" state. Thus, in the exemplary embodiment of Fig. 4, it may be possible to actually use only aggregator drivers 211, 213, 214, 215 and 217, which is less than a one-to-one ratio with the number of ports on network switch 240. However, it may also be possible that through dynamic changes in the linking, that aggregator drivers 212, 216 and 218 may become associated with one or more of physical ports 241-248 during the operation of the network.

Level 210 aggregator drivers 211-218 may also be bound in the downward direction to multiple level 220 network drivers. For example, aggregator driver 201 is bound in the downward direction to aggregation port drivers 221 and 222, and aggregator driver 215 is bound in the downward direction to aggregation port drivers 225-227. However, level 220 aggregation port drivers 221-228 may only bind in the upward direction to a single level 210 aggregator driver. For example, aggregation port driver 221 is bound to aggregator driver 211, and therefore, may not bind to any other aggregator driver. As will be described in greater detail below, aggregation port drivers may dynamically change the aggregator driver to which they are bound, but each will be bound to only one aggregator driver at a time.

The level 220 aggregation port drivers 221-228 represent the actual physical ports of network switch 240, and, therefore there is a one-to-one ratio between the level 220 aggregation port drivers 221-228 and physical ports 241-248 of network switch 240. Similarly, there is a one-to-one ration between the level 220 aggregation port drivers 221-228 and the level 230

hardware device drivers 231-238. However, the level 220 aggregation port drivers 221-228 do not directly touch the physical ports, but rather act to abstract the actual level 230 hardware device drivers 231-238 in order to isolate level 220 and the other levels of network drivers, levels 210 and 200 from the hardware device. The purpose of isolating levels 200-220 from the actual hardware is that each of the network drivers in these levels, e.g., interface network drivers 201-203, aggregator drivers 211-218 and aggregation port drivers 221-228, may be designed without an understanding of any specific hardware device. This generic design means that if a hardware device is changed or a level 230 hardware device driver is changed, there is no need to make any changes to the other levels of network drivers. Additionally, the abstraction allows the developer of level 230 hardware device drivers 231-238 to develop drivers that address the specific capabilities of the hardware device without the need to understand how to interface with the various network software layers.

Level 230 hardware device drivers 231-238 directly control and pass or receive data packets from the hardware device, which in this example, is network switch 240. Hardware device drivers 231-238 bind in the upward direction to the level 220 aggregation port drivers 221-228 on a one-to-one basis. Similarly, hardware device drivers 231-238 bind in the downward direction directly to ports 241-248 of network switch 240 on a one-to-one basis. For example, if network switch 240 had 12 ports, there would be 12 hardware device drivers in level 230 and twelve aggregation port drivers in level 220. Generally, hardware device drivers 231-238 provide services such as the initialization of hardware and hardware support tasks, hardware management and data packet transfer. However, the specific tasks accomplished by hardware device drivers 231-238 are highly dependent on the underlying network hardware device. For example, when initializing a hardware device, hardware device drivers 231-238 may set hardware control registers or support hardware interrupts and vectors. As a further example, in the case of network switch 240, hardware device drivers 231-238 may be responsible for switch and port statistic retrieval and management. Other network driver levels may also be responsible for port statistics, for example, level 220 aggregation port drivers 221-228 may poll the port statistics from the hardware device drivers 231-238 and add delta values to counters that may be employed in level 200 and 210 network drivers. Those skilled in the art will understand that hardware device drivers 231-238 may be

specific to the particular hardware device they are controlling or may be a generic hardware device driver provided with the other network drivers. For example, hardware device drivers 231-238 may have been developed by the manufacturer of network switch 240 to most efficiently use all the capabilities of network switch 240. On the other hand, hardware device drivers 231-238 may be a generic hardware device driver provided with the network software. In either case, it does not effect the other network drivers, because, as described above, the level 200-220 network drivers are generic and independent of the hardware and the level 230 hardware device drivers.

The multiple levels of network drivers allow for dynamic changes in the membership of VLANs and for membership changes in link aggregations. Fig. 4 may be considered the initial static set up of network driver levels 200-230 for the eight port network switch 240. In this initial set up, there is a one-to-one correspondence between physical ports 241-248 of switch 240 and the level 230 hardware device drivers 231-238. Each hardware device driver 231-238 is bound to and responsible for controlling one of ports 241-248 of switch 240. In the upward direction, each hardware device driver 231-238 is bound on a one-to-one basis to aggregation port drivers 221-228. These links between physical ports 241-248 of switch 240, hardware device drivers 231-238 and aggregation port drivers 221-228 will not change after initialization. For example, during initialization, physical port 241 was bound to hardware device driver 231 which was bound to aggregation port driver 221. Regardless of any dynamic changes that will be made during operation, these links will remain.

As previously described, link aggregation allows multiple physical links to be combined to operate as a single logical link. Viewing the links in Fig. 4, it should become apparent that some groups of physical ports 241-248 of switch 240 are aggregated into single logical links. Specifically, physical ports 241-242 are aggregated into a first logical link and physical ports 245, 246 and 247 are aggregated into a second logical link. Those skilled in the art will understand that the connections through ports 243, 244 and 248 are still considered logical links, even though they are single logical links. These single logical links may still be referred to as an aggregated link because there is still a connectivity to the upper level layers. The following examples will discuss a link that has multiple aggregated ports, however, the

present invention is not limited to such an arrangement.

Using the example of link aggregation between physical ports 241 and 242, Fig. 4 shows that the links from both of physical ports 241 and 242 converge at aggregator driver 211 which may be considered the logical port of the logical link. Aggregator driver 211 appears as a single logical port to the entire system above it, including interface network driver 201 and upper level applications, for example, STP and GARP. Thus, whether an aggregator driver is linked to zero, one or multiple aggregation port drivers, it appears as a single logical port to the system above it. For example, both aggregator drivers 211 and 213 appear as a single logical port to interface network driver 201, even though aggregator driver 211 is associated with physical ports 241 and 242, while aggregator driver 213 is associated only with a single physical port 243.

In the initialized state of Fig. 4, the system sees the eight physical ports 241-248 of network switch 240 as five logical ports that are "on" or "up" in the form of aggregator drivers 211, 213, 214, 215 and 217, while aggregator drivers 212, 216 and 218 are logical ports that are considered "off" or "down." The multiple levels of network drivers allow the link aggregation to change in a manner that is transparent to the upper level applications. For example, Fig. 6 shows an alternate collection of exemplary links between layer 210 aggregator drivers 211-218 and layer 220 aggregation port drivers 221-228. Fig. 6 shows a configuration of links that is changed from the links at system initialization such as may happen with dynamic configuration as the network is operating. The changes in the link configuration from Fig. 4 to Fig. 6 are as follows: aggregation port driver 222 has changed its binding from aggregator driver 211 to aggregator driver 213 and aggregation port driver 227 has changed its binding from aggregator driver 215 to aggregator driver 217. As described above, each aggregation port driver 221-228 is bound in the upward direction to only one aggregator driver 211-218, however the specific bindings have changed. These changes in the bindings result in physical ports 242 and 243, physical ports 245 and 246, and physical ports 247 and 248 being aggregated. However, level 200 interface network drivers 201-203 and the upper level layers and applications do not see any change in the system. Interface network drivers 201-203 and the upper level layers and applications still see the five logical

ports that are up in the form of aggregator drivers 211, 213, 214, 215 and 217. Although there has been a major change in the manner in which physical ports 241-248 are aggregated, this change is completely transparent to interface network drivers 201-203 and the upper level layers and applications. Therefore, the change in link aggregation does not require any resulting changes in these upper level layers or applications.

Similarly, there may come a time when, for example, aggregation port driver 222 changes its link from aggregator driver 211 to aggregator driver 212. Again, there is a major change in the link aggregation, but the only change seen by the upper level layers and applications is that aggregator driver 212 is now up instead of down. In terms of the VLAN, there are no changes seen by the upper level layers because aggregator driver 212 was receiving the VLAN(A) traffic, but was discarding it because it had nowhere to pass the information. Now, aggregator driver 212 may pass the information through to port 242 for physical transmission.

The changes in the links between level 220 aggregation port drivers 221-228 and level 210 aggregator drivers 211-218 may be performed manually or may occur automatically based on a predefined criteria. Referring back to Fig. 4, the initial system set up may be based on the characteristics or parameter settings of the network drivers, for example, the default settings for various parameters. For example, each of the aggregator drivers 211-218 and aggregation port drivers 221-228 has certain parameter settings. Based on these parameter settings, links are established between like aggregation port drivers and aggregator drivers. In Fig. 4, the parameter settings of aggregation port drivers 221 and 222 match the parameter settings of aggregator driver 211, and therefore links are established between these drivers. The parameter settings may include characteristics such as media type, duplex port speed, link state (up or down), etc. During network operation, there may be some change to the parameter settings of the aggregation port drivers that cause it to no longer match the parameter settings of the aggregator driver to which it is linked. When this occurs, the link between the aggregation port driver and the aggregator driver is broken. For example, the initial parameter setting for port speed of aggregator driver 211 and aggregation port drivers 221 and 222 may have been 100 Mbps. However, during network operation the parameter setting for port speed of aggregation port driver 222 may be changed to 10Mbps. This change

in parameter setting will cause the link between aggregation port driver 222 and aggregator 211 to be broken. Referring back to Fig. 6, a network administrator may manually link aggregation port driver 222 to aggregator driver 213 when it becomes disconnected from aggregator driver 211. In the alternative, aggregation port driver 222 may be programmed to search automatically for another aggregator that has the same parameter settings and bind to that aggregator. To follow through with the example started above, once the port speed parameter setting of aggregation port driver 222 changes and it is disconnected from aggregator driver 211 it begins to search for another aggregator driver having the same parameter settings. This search reveals that aggregator driver 213 has the same parameter settings, including the 10 Mbps port speed, as aggregation port driver 222. Therefore, aggregation port driver 222 binds to aggregator driver 213.

It should be noted that the actual physical port which transmits the data packet when multiple physical ports are aggregated may be determined either in software or in hardware. The data packet is addressed for a specific destination, but when physical ports are aggregated together, it means that each of these ports lead to the same logical destination, there are just one or more parallel paths to that logical destination. In a hardware implementation, the data packet is passed to the hardware level and functions within the hardware make a determination as to which port the data packet should be transmitted. In contrast, a software implementation means that the physical port for transmission is determined by software. Those skilled in the art will understand that there are many methods to make such a determination based on many factors, but the key factor is that the manner should be deterministic.

Referring to Fig. 4 an example of data packet flow will be given for both a hardware and software implementation of physical port selection. Physical ports 245, 246 and 247 are aggregated forming three parallel paths through which data packets may flow. When a transmission data packet reaches aggregator driver 215, the data packet may continue down to any one of aggregation port drivers 225, 226 or 227. In the hardware implementation, aggregator driver 215 has no function to determine which aggregation port driver path is selected because network switch 240 will ultimately make the determination as to which port

will transmit the data packet. Therefore, aggregator driver 215 may transmit every data packet down the same path, for example, to aggregation port driver 225 through to hardware device driver 235 and finally to physical port 245. When the data packet arrives at physical port 245, network switch 240 hardware will make the determination as to which of physical ports 245, 246 or 247 should transmit the data packet. In the software implementation, aggregator driver 215 will make the determination as to which port of physical ports 245, 246 or 247 should transmit the data packet. As described above, the determination will be made based on a deterministic criteria with a view to achieve load balancing between the three physical ports, meaning that no one of physical port 245, 246 and 247 becomes overloaded or underutilized. Based on this determination, aggregator driver 215 will transmit the data packet to the selected one of aggregation port drivers 225, 226 or 227 so the packet will arrive at the correct physical port for transmission.

In addition to the dynamic reconfiguring of link aggregations, the multiple levels of network drivers also allow for the dynamic reconfiguration of VLANs. In Fig. 4, physical ports 241-248 of network switch 240 are members of three VLANs, VLAN(A), VLAN(B) and VLAN(C). Thus, there are three interface network drivers 201-203, because as described above, there is a one-to-one ratio between VLANs and interface network drivers. In Fig. 4, the logical ports, aggregator drivers 211-218, bind to interface network drivers 201-203 based on VLAN membership. For example, aggregator driver 211 is bound to interface network driver 201 because the logical port, and consequently the underlying physical ports 241 and 242, are members of VLAN(A). Aggregator driver 214 is bound to both interface network driver 201 and 202 because the logical port, and consequently the underlying physical port 244, are members of VLAN(A) and VLAN(B). As shown by this example, a logical port (aggregator drivers 211-218) may be members of multiple VLANs. It may also be possible that a logical port is not a member of any VLAN. This may be altered during network operation by dynamic addition to a VLAN, but VLAN membership is not required.

In Fig. 4, aggregator driver 215 (logical port) is bound to interface network driver 203 making it and underlying physical ports 245, 246 and 247 members of VLAN(C). If, for example, physical ports 245, 246 and 247 were connected to a personal computer (PC), the initial set

up of the network shown in Fig. 4 would mean that this PC is only a member of VLAN(C). During the operation of the network, the user of the PC may desire to also become a member of VLAN(A) and VLAN(B) so she can send and receive network traffic on those VLANs. In this case, aggregator driver 215 may dynamically bind to interface network drivers 201 and 202 also, meaning that aggregator driver 215 is now a member of all three VLANs, and will receive the network traffic for each VLAN. To join these additional VLANs, the user or a network administrator may change a parameter of aggregator driver 215. This parameter change may result in a message or functional interface to the interface network drivers 201 and 202 associated with the VLAN that aggregator driver 215 wishes to join. After this message is sent, aggregator driver 215 binds to interface network drivers 201and 202. Similarly, aggregator driver 215 may also relinquish membership in a VLAN by breaking the link between it and the interface network driver for that particular VLAN. But, once again, regardless of the number and type of dynamic changes to the VLAN configuration, it remains completely transparent to the upper level layers and applications. It should be noted that there may be some failure states of the VLAN or aggregation links that may not be transparent to the upper level layers. For example, if an aggregation link has three parallel paths between network devices and all three paths fail, the data packet will not be transmitted to the receiving network device. In such a case, the upper level layers may be signaled that a transmission failure occurred. However, in normal operation of the network, dynamic changes will remain transparent to the upper level layers.

Fig. 7 shows a block diagram of an exemplary embodiment of stacked network drivers implemented on 8 port network switch 300 including upper level layers and applications. Exemplary network switch 300 employs protocols that are part of the Internet Protocol (IP) suite of protocols. Network switch 300 has 8 physical ports 391-398 connected to hardware device drivers 381-388. This one-to-one correspondence between hardware device drivers and physical ports means that when a hardware device driver receives a data packet, it always knows to which physical port the data packet should be sent, *i.e.*, the physical port to which it is connected. Hardware device drivers 381-388 are bound in the upward direction to aggregation port drivers 371-378 which are bound in the upward direction to aggregator drivers 361-368 which are bound in the upward direction to interface port drivers 351-354.

Each of the network drivers has a structure that describes the driver to which it is bound in the upward and downward direction. For example, the structure of aggregator driver 363 describes parameters of aggregation port driver 373 because of the binding in the downward direction. The structure may include parameter information such as media type and port speed. Aggregator driver 363 also contains information about interface network drivers 351 and 352 because of the binding in the upward direction. A simple change in the contents of the structure of any of the network drivers may result in new network driver configuration, but this change will remain completely transparent to the upper level layers and applications of the network software. For example, the network administrator may change the parameters of aggregation port driver 375 so that the parameters no longer match the parameters of aggregator driver 364 and the link will be terminated, thus the information contained in aggregation port driver 375 about aggregator driver 364 may be purged, and vice versa. The new parameters may of aggregation port driver 375 may match aggregator driver 366 and therefore, these two network drivers may be bound. In the binding process, aggregation port driver 375 and aggregator driver 366 may exchange and store information in their respective structures about the network driver(s) to which they are now bound.

Fig. 7 also shows management applications such as web server 301 and Telnet 305. The RIP 315 (Routing Information Protocol) is an IP routing protocol that is based on the simple distance vector routing algorithm. RIP 315 is an interior gateway protocol that is UDP (User Datagram Protocol) based and directly updates the IP stack routing table (not shown). TCP 310 (Transmission Control Protocol), UDP 320 and OSPF 325 (Open Shortest Path First) run directly over IP layer 330. OSPF 325 is a link state routing protocol that is IP based and directly updates the IP stack routing table (not shown). IP layer 330 directly interfaces with multiplexer 345 which abstracts the protocols from the network drivers. STP 335 and GARP 340 are applications which perform specific network functions. As described above, STP 335 and GARP 340 may also bind directly to aggregator drivers 361-368. Note that regardless of how the links between the network drivers are dynamically changed, there is no effect on the upper level protocols or applications.

Fig. 8 shows an exemplary process flow for a data packet through the upper level protocols and network drivers using the exemplary arrangement shown in Fig. 7. In step 400, web server 300 receives a request for a particular web page. Web server 300 sends the appropriate data packets to TCP 310 in step 405. TCP 310 adds any desired information to the data packet and then passes the data packet to IP 330 in step 410. Those skilled in the art will understand that each layer may perfom other processing steps in addition to the simple addition of data, for example, the data may be "chopped" to the proper packet size, a header structure may be added, etc. The use of the term adding information to the data packet in this description also includes this other necessary processing by the different layers. In step 415, IP adds additional information, including IP addressing information, to the data packet. As described above, IP 330 understands the subnet on which the data packet is to be transmitted, but has no understanding of VLANs or physical ports. There is no mechanism that allows IP 330 to pass the data packet directly to hardware device drivers 381-388 or even to the logical ports, aggregator drivers 361-368, because IP 330 does not know which physical port or ports are associated with the IP address. The data packets will then be passed to multiplexer 345 in step 415, which will, in turn pass the data packet to the correct one of interface network driver 351-354 based the connected subnets in step 420. Once the data packet is at the interface network driver level, it has arrived in the correct VLAN, because, as described above, there is a one-to-one correlation between VLANs and interface network drivers.

The structures of interface network drivers 351-354 may also contain information about each of connected aggregator drivers 361-368. Based on this information, interface network drivers 351-354 pass the data packet to the correct one or more of aggregator drivers 361-368 in step 425. In step 430, aggregator drivers 361-368 pass the data packet to aggregation port drivers 371-378. Similar to the previous step, the structures of aggregator drivers contain information about each of connected aggregation port drivers 371-378 and based on this information the data packet is passed to the correct one of aggregation port drivers 371-378. In steps 435 and 440 the data packet continues to hardware device drivers 381-388 to physical ports 391-398 and is finally sent out of the physical port in step 445.

As can be seen from the previously described steps, the upper level protocols have no connection to the network drivers except that IP layer 330 addresses data packets based on subnets and interface network drivers 351-354 are connected to subnets. However, the subnets have no direct correlation to VLANs, physical or logical ports. Therefore, membership in the underlying VLANs or aggregations of physical links into logical ports may change without having any effect on the upper level protocols and applications. The stacked network drivers allow for these changes to the VLAN membership and link aggregations without effecting the upper level protocols and applications.

In the preceding specification, the present invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereunto without departing from the broadest spirit and scope of the present invention as set forth in the claims that follow. The specification and drawings are accordingly to be regarded in an illustrative rather than restrictive sense.

**Claims**

What is Claimed is:

1. A software package for operating on a network hardware device having a plurality of physical ports and at least one networking protocol to transmit and receive data packets between a plurality of VLANs, comprising:

a plurality of first network drivers communicatively coupled to the plurality of physical ports of the network hardware device;

a plurality of second network drivers selectively communicatively coupleable to a first one of the plurality of first network drivers and selectively communicatively coupleable to a second one of the plurality of first network drivers; and

a plurality of third network drivers, each third network driver being associated with a corresponding one of the plurality of VLANs, wherein each of the third network drivers is selectively communicatively coupleable to at least one of the second network drivers that is a member of the corresponding one of the plurality of VLANs, and wherein a data packet for transmission by the network hardware device travels from one of the third network drivers to the corresponding communicatively coupled second network drivers, from the communicatively coupled second network drivers to the first network drivers communicatively coupled thereto and from the communicatively coupled first network drivers to the physical ports to which the communicatively coupled

first network drivers are coupled.

2. The software package of claim 1, wherein data packets received by the network hardware device travel from communicatively coupled ones of the physical ports to the first network drivers communicatively coupled thereto, from the communicatively coupled first network drivers to the second network drivers communicatively coupled thereto, and from the communicatively coupled second network drivers to the third network drivers communicatively coupled thereto.

3. The software package of claim 1, further comprising:

at least one network application communicatively coupleable with the second network drivers for communicating control information with the network hardware device.

4. The software package of claim 1, further comprising:

at least one upper level network layer having a plurality of subnets, wherein each of the third network drivers is selectively communicatively coupleable with each of the plurality of subnets based on a correlation between each of the plurality of subnets and the VLAN corresponding to each of the third network drivers.

5. The software package of claim 4, wherein data packets are passed between the third network drivers and the subnets communicatively coupled thereto.

6. The software package of claim 1, wherein the second network drivers dynamically change the selectively communicatively coupling to the first one of the first network drivers and the second one of the first network drivers.

7. The software package of claim 1, wherein each of the third network drivers dynamically changes the selectively communicatively coupling to the at least one of the second network drivers based on changes in VLAN membership of the at least one of the second network drivers.

8. The software package of claim 1, further comprising:

a plurality of fourth network drivers communicatively coupled to the physical ports and the first network drivers, wherein data packets traveling between the communicatively coupled physical ports and first network drivers pass through the fourth network drivers communicatively coupled thereto.

9. A software package for operating on a network including a plurality of network hardware devices having at least one networking protocol to transmit and receive data packets over the network, comprising:

at least one upper level network layer;

a plurality of first network drivers communicatively coupleable to a plurality of physical ports on a first network hardware device;

a plurality of second network drivers selectively communicatively coupleable to a first one of the first network drivers and a second one of the first network drivers; and

a plurality of third network drivers, each of which is selectively communicatively coupleable to at least one of the second of network drivers, wherein the third network drivers receive data packets for transmission containing addressing information from the at least one upper level network layer and send the data packets to selected ones of the second network drivers based on the addressing information, the selected ones of the second network drivers sending the data packets to the first network drivers communicatively coupled thereto, the communicatively coupled first network drivers sending the data packets to the physical ports communicatively coupled thereto and the communicatively coupled physical ports transmitting the data packets to a second network hardware device.

10. The software package of claim 9, wherein each of the third network drivers is a member of a corresponding VLAN and each of the third network drivers is selectively communicatively coupleable to each of the plurality of subnets based on a correlation between each of the plurality of subnets and the VLAN corresponding to each of the third network drivers.

11. The software package of claim 9, wherein data packets received by the physical ports of the first network hardware device travel from the physical ports to the first network drivers communicatively coupled thereto, from the communicatively coupled first network drivers to the second network drivers communicatively coupled thereto, from the communicatively coupled second network drivers to the third network drivers communicatively coupled thereto.

12. The software package of claim 9, wherein the second network drivers send the transmission data packets to one of the first one of the first network drivers and the second one of the first network drivers.

13. The software package of claim 9, wherein each of the third network drivers includes a first set of parameters and each of the second network drivers includes a second set of parameters, and each of the third network drivers selectively communicatively couples to the at least one second network drivers based on a common set of the first and second set of parameters.

14. The software package of claim 13, wherein each of the third network drivers selectively communicatively decouples with the at least one second network drivers based on a change to one of the first and second set of parameters.

15. The software package of claim 9, wherein each of the second network drivers includes a first set of parameters and each of the first network drivers includes

a second set of parameters, and each of the second network drivers selectively communicatively couples to the first one of the first network drivers based on a common set of the first and second set of parameters.

16. The software package of claim 15, wherein each of the second network drivers selectively communicatively decouples with the first one of the first network drivers based on a change to one of the first and second set of parameters.

17. A hardware device, comprising:

a plurality of physical ports for receiving and transmitting data packets, wherein a first one of the physical ports is a member of one of a first VLAN and a second VLAN and a second one of the physical ports is a member of one of the first VLAN and the second VLAN;

a plurality of first network drivers communicatively coupleable to a corresponding one of the plurality of physical ports;

a plurality of second network drivers, each of the second network drivers being selectively communicatively coupleable to a first one of the first plurality of network drivers and a second one of the first plurality of network drivers; and

a plurality of third network drivers, wherein a first one of the third

network drivers is a member of the first VLAN and a second one of the third network drivers is a member of the second VLAN, the first one of the third network drivers being communicatively coupled to the one of the first and second physical ports that is a member of the first VLAN, the second one of the third network drivers being communicatively coupled to the one of the first and second physical ports that is a member of the second VLAN.

18. A network system, comprising:

a plurality of interconnected network hardware devices for communicating data packets;

a plurality of first network drivers communicatively coupleable to a plurality of physical ports on a first network hardware device;

a plurality of second network drivers selectively communicatively coupleable to a first one of the first network drivers and a second one of the first network drivers; and

a plurality of third network drivers, each of which is selectively communicatively coupleable to at least one of the second network drivers, wherein each of the third network drivers send data packets to the second network drivers coupled thereto, and wherein the second network drivers send the data packets to the first network drivers coupled thereto, the first network

drivers sending the data packets to the physical ports coupled thereto, the data packets being transmitted from the physical ports to network hardware devices coupled thereto.

19. The network system of claim 18, further comprising:

at least one network application communicatively coupleable to the second network drivers for communicating control information to the first network hardware device.

20. The network system of claim 18, further comprising:

at least one networking protocol to facilitate communication between network hardware devices.

21. The network system of claim 20, wherein the at least one networking protocol includes a TCP/IP protocol suite.

22. The network system of claim 18, wherein each of the third network drivers dynamically changes the selectively communicatively coupling to the at least one of the second network drivers.

23. The network system of claim 18, the first one of the network hardware devices being a router.

24. The network system of claim 18, the first one of the network hardware devices being a network switch.

25. The network system of claim 18, the first one of the network hardware devices being a network interface card.

26. The network system of claim 18, the first one of the network hardware devices being an internet appliance.

27. The network system of claim 18, the first one of the network hardware devices being a personal computer.

28. A method for communicating data packets by a network hardware device over a network, comprising the steps of:

addressing data packets to include information for transmission;

sending each of the data packets to a first one of a plurality of first network drivers based on the addressing information in the data packet;

reading the data packet addressing information in the first one of the first network drivers;

adding additional addressing information to the data packet in the first one of the first network drivers;

sending the data packet to at least one of a plurality of second network drivers communicatively coupleable to the first one of the first network drivers based on the addressing information in the data packet;

sending the data packet to at least one of a plurality of third network drivers communicatively coupleable to the at least one of the second network drivers based on the addressing information in the data packet;

sending the data packet to at least one of a plurality of physical ports communicatively coupleable to the at least one third network driver based on the addressing information in the data packet; and

transmitting the data packet to the network via the at least one of the plurality of physical ports.

29. A method of connecting network drivers in multiple levels, comprising the steps of:

selectively communicatively coupling a first one of a plurality of first network drivers to at least one of a plurality of second network drivers;

selectively communicatively coupling the at least one of the second network drivers to at least one of a plurality of third network drivers; and

communicatively coupling the at least one third network driver to a first one of a plurality of physical ports.

30. The method of claim 29, wherein the first one of the first network drivers is a member of a first VLAN and the at least one second network drivers communicatively couples to the first one of the first network drivers based on the membership in the first VLAN.

31. The method of claim 29, further comprising the step of:

decoupling the first one of the first network drivers from the at least one second network drivers.

32. The method of claim 29, further comprising the step of:

decoupling the at least one second network drivers from the at least one third network drivers.

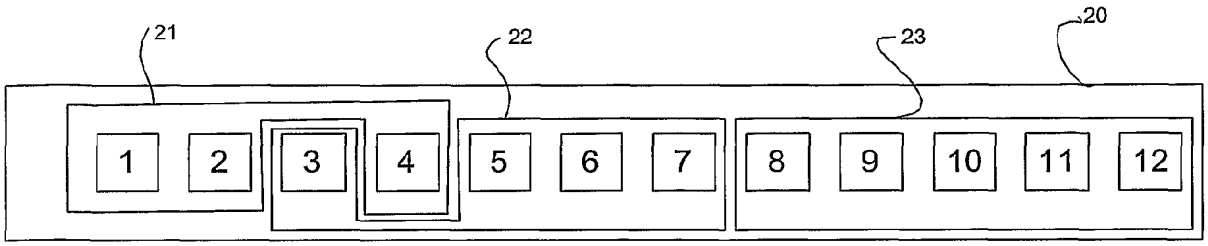
21
22
23
20
1
2
3
4
5
6
7
8
9
10
11
12

Fig. 1

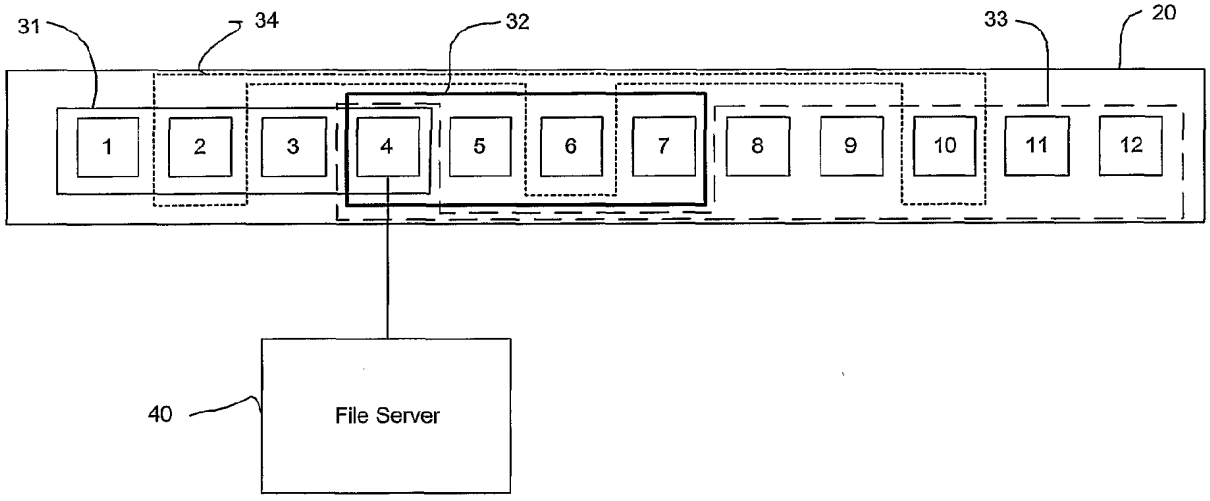
31
34
32
33
20
1
2
3
4
5
6
7
8
9
10
11
12
40
File Server

Fig. 2

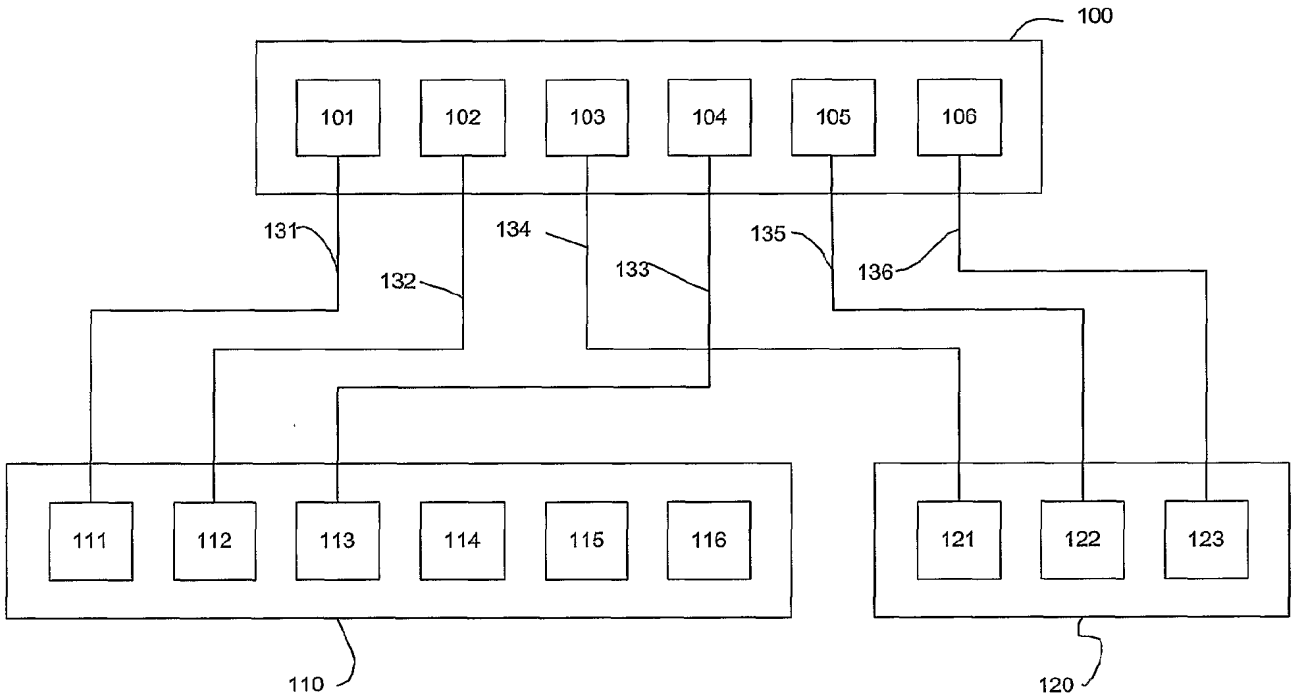
100
101
102
103
104
105
106
131
132
133
134
135
136
111
112
113
114
115
116
121
122
123
110
120

Fig. 3

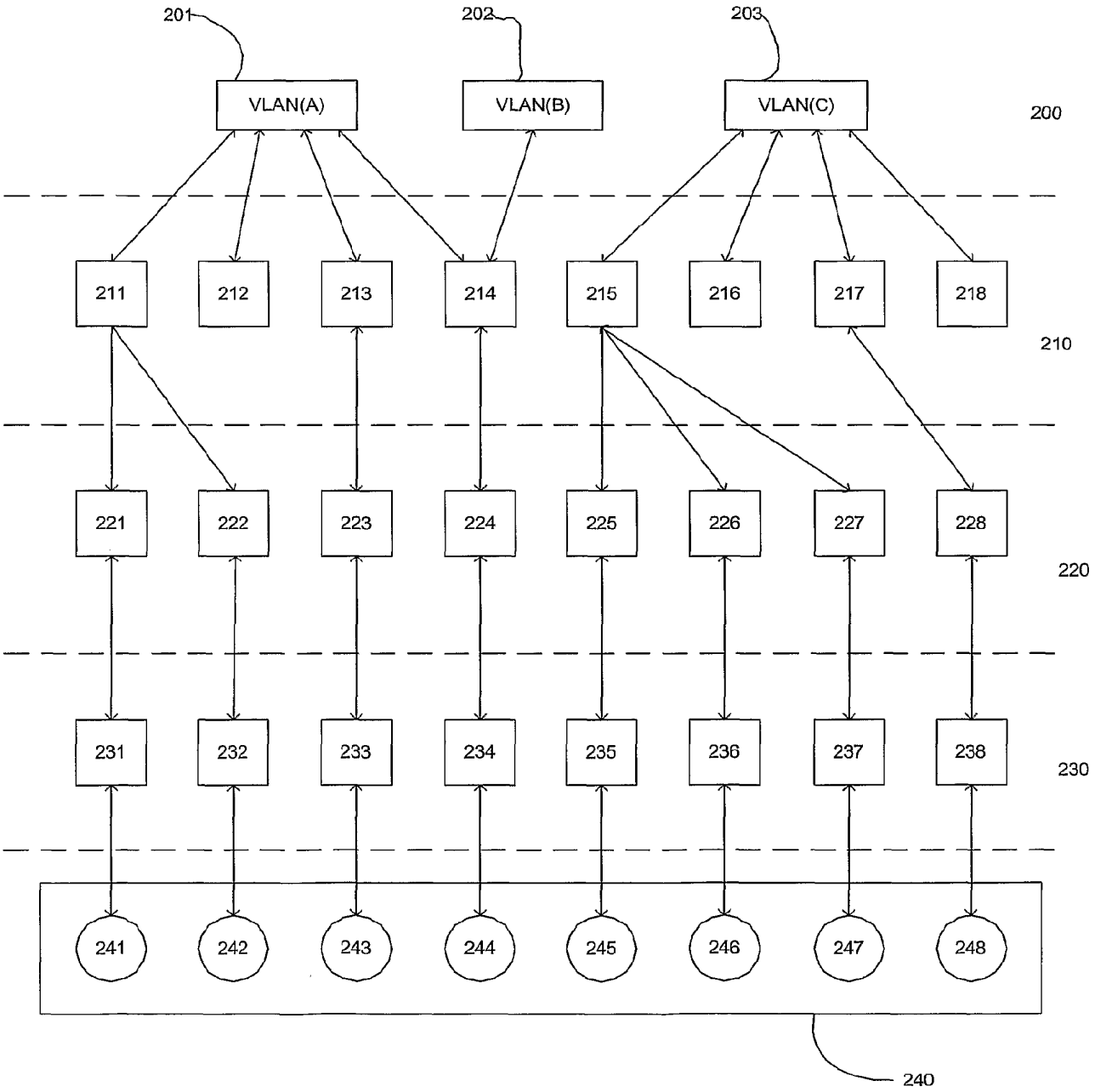
201
202
203
VLAN(A)
VLAN(B)
VLAN(C)
200
211
212
213
214
215
216
217
218
210
221
222
223
224
225
226
227
228
220
231
232
233
234
235
236
237
238
230
241
242
243
244
245
246
247
248
240

Fig. 4

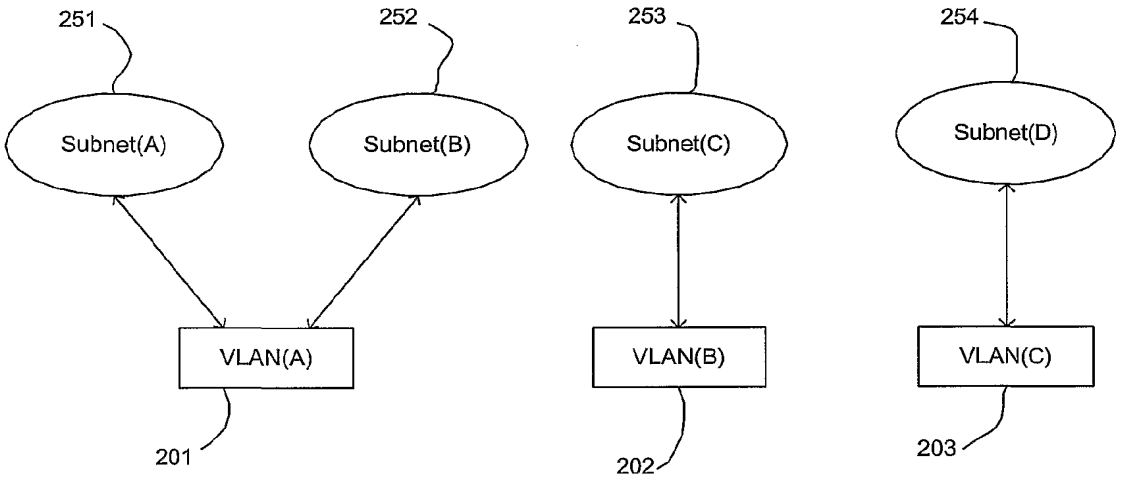
251
252
253
254
Subnet(A)
Subnet(B)
Subnet(C)
Subnet(D)
VLAN(A)
VLAN(B)
VLAN(C)
201
202
203

Fig. 5

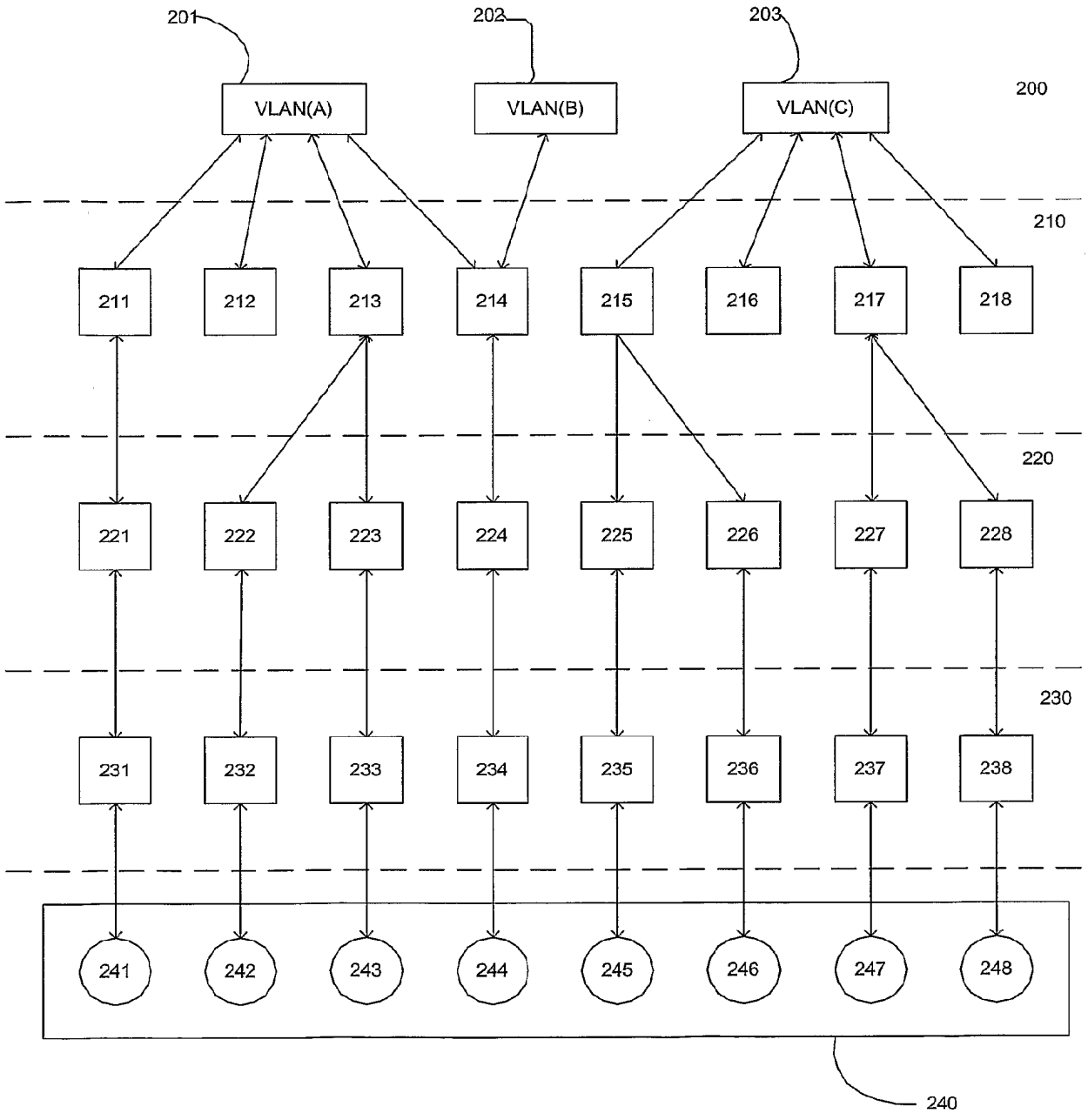
201
202
203
VLAN(A)
VLAN(B)
VLAN(C)
200
210
211
212
213
214
215
216
217
218
220
221
222
223
224
225
226
227
228
230
231
232
233
234
235
236
237
238
241
242
243
244
245
246
247
248
240

Fig. 6

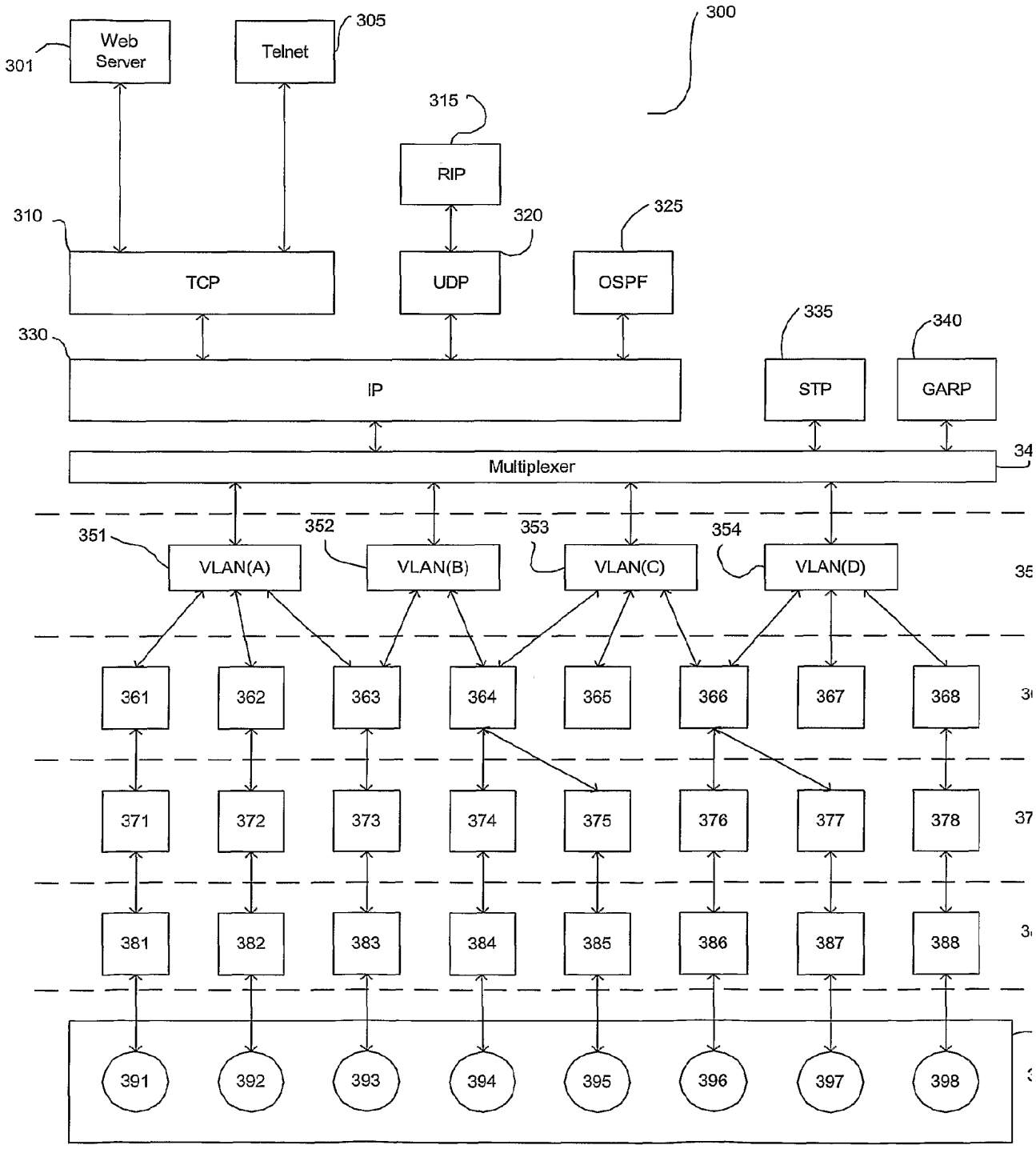
300
Web Server
301
Telnet
305
RIP
315
TCP
310
UDP
320
OSPF
325
IP
330
STP
335
GARP
340
Multiplexer
34
VLAN(A)
351
VLAN(B)
352
VLAN(C)
353
VLAN(D)
354
361
362
363
364
365
366
367
368
371
372
373
374
375
376
377
378
381
382
383
384
385
386
387
388
391
392
393
394
395
396
397
398

Fig. 7

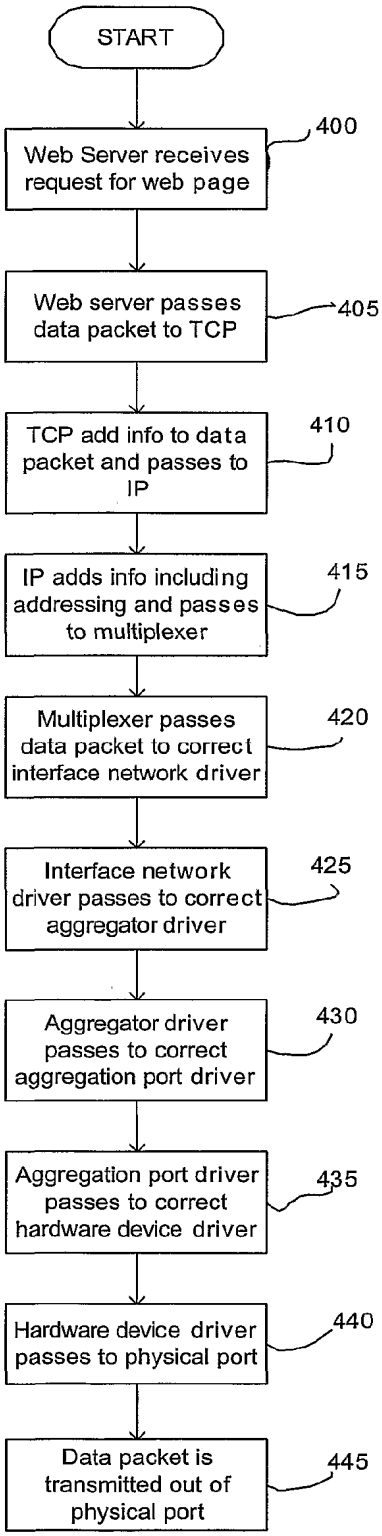
START
Web Server receives request for web page
400
Web server passes data packet to TCP
405
TCP add info to data packet and passes to IP
410
IP adds info including addressing and passes to multiplexer
415
Multiplexer passes data packet to correct interface network driver
420
Interface network driver passes to correct aggregator driver
425
Aggregator driver passes to correct aggregation port driver
430
Aggregation port driver passes to correct hardware device driver
435
Hardware device driver passes to physical port
440
Data packet is transmitted out of physical port
445

Fig. 8